# Advanced Topics in Cryptography

## Lecture 9: Pairing based cryptography, Identity based encryption.

Benny Pinkas

# Related papers

– Lecture notes from MIT
  http://crypto.csail.mit.edu/classes/6.876/lecture-notes.html

– Clifford Cocks , An Identity Based Encryption Scheme based on Quadratic Residues.
  http://www.cesg.gov.uk/site/ast/idpkc/media/ciren.pdf

# Bilinear maps: motivation

- Bilinear maps are the tool of pairing-based cryptography
  - First major application: an efficient identity-based encryption scheme (2001).
  - Manu more applications.

- What can they do?
  - Establish relationships between cryptographic groups
  - Make DDH easy in one of the groups
  - Enable to solve the CDH once

# Bilinear Maps

- Let G, $G_t$ be cyclic groups of the same order
- A bilinear map from $G \times G$ to $G_t$ is a function $e: G \times G \rightarrow G_t$, such that
  - $\forall$ u,v $\in$ G, a,b$\in$ Z,

    $e(u^a,v^b) = (e(u,v))^{ab}$
- This is true if and only if $\forall\ u_1,u_2,v_1,v_2 \in G$
  - $e(u_1+u_2,v_1) = e(u_1,v_1)\cdot e(u_2,v_1)$
  - $e(u_1,v_1+v_2) = e(u_1,v_1)\cdot e(u_1,v_2)$
- A bilinear map is called a pairing since it associates pairs of elements from G with an element in $G_t$.

# Admissible bilinear maps

- A bilinear map can be *degenerate*: map everything to 1, and therefore $e(u^a,v^b) = 1 = (e(u,v))^{ab} = 1^{ab}$

- Let g,g' be generators of G.
- A bilinear map is called *admissible* if *e(g,g')* generates $G_t$, and *e* is efficiently computable.

- These are the only maps we care about.
- G and $G_t$ have the same order G(g,1) generates $G_t$.
- If $G=G_t$ then we get a very powerful primitive.
  - But it's unknown how to construct such a pairing

# Another notation

- It is common to use an *additive* notation for the group G. Namely,
  - The operation in G is +
  - 1 is a generator of G
  - The discrete log problem means that given (g, a·g) it is hard to find a.

- We will use the multiplicative notation

# Implications to the Discrete Log problem

- The discrete log problem in G is no harder than the discrete log problem in $G_t$.
- Our input is $(g,g^a)$ from G, for a random a, and we need to find a.
- Suppose that it is easy to compute discrete logarithms in $G_t$. We work as follows:
  - $g_t = e(g,g)$
  - $p = e(g,g^a)$
  - Find the discrete log (in $G_t$) of p to the base $g_t$
- This works since $e(g,g^a) = e(g,g)^a$

# Implications to the Decisional Diffie-Hellman problem (DDH)

- The DDH problem in G is easy.
- Our task is to distinguish between $\langle g,g^a,g^b,g^{ab}\rangle$, and $\langle g,g^a,g^b,g^c\rangle$, for random a,b,c.
- The distinguisher is given $\langle P,A,B,C\rangle$
  - It computes $v_1$=e(A,B) and $v_2$=e(P,C)
  - It declares "DDH" if and only if $v_1=v_2$

- Indeed, If $C=P^{ab}$ then $e(A,B)=e(g^a,g^b)=e(g,g)^{ab}=e(g,g^{ab})$
  - And since the mapping *e* is non-degenerate, this equality happens if and only if c=ab.
- Note that we can only solve the DDH in G, and therefore we can only solve it *once*.

# Diffie-Hellman implications

- What about the CDH (Computation Diffie-Hellman) problem?
  - Bilinear maps are not known to be useful for solving the CDH. Therefore this problem might still be hard in G.

- A group G is called a gap Diffie-Hellman group (GDH) if the DDH is easy in G but the CDH is hard
  - The definition is independent of the use of bilinear maps
  - But bilinear maps enable to construct gaps groups

# What groups to use?

- Typically G is an elliptic curve
  - An elliptic curve is defined by $y^2=x^3+1$ over a finite field $F_p$.
  - There are many types of curves
- The group $G_t$ is normally a finite field
- The bilinear maps are usually the Weil or Tate pairings
  - Pretty complicated
  - Overhead of the same order as that of exponentiation
- We don’t need to understand the details of implementing bilinear pairings in order to use them.

# New problems – cryptographic assumptions

- In order to design new cryptographic protocols based on pairings, we need to make new assumptions
  - Bilinear Diffie-Hellman: given $\langle g, g^a, g^b, g^{ab} \rangle$ it is hard to compute $e(g,g)^{abc}$ (a “three-way Diffie-Hellman, but the result is in $G_t$).
  - Decisional Bilinear Diffie-Hellman: it is hard to distinguish $\langle g, g^a, g^b, g^{ab} \rangle$ from $\langle g, g^a, g^b, g^c \rangle$
  - Similar assumptions when the mapping is $e: G_1 \times G_2 \rightarrow G_t$

# Intuition

- Whay are bilinear maps so useful?
  - They enable to solve the DDH problem, but only once!
  - The solution is easy if we have elements in G. But the solution itself generates elements in G_t for which cannot apply the mapping.
  - This level of power enables to construct cryptographic protocols, but is not enough for the adversary to attack the system.

# Joux's 3-party Diffie-Hellman protocol

- The goal: let three parties decide on a key using DH
- Can easily do it with in two rounds. We want to do it in a single round
- Let G be a group in which DH is hard, and g a generator. $e:G\times G \rightarrow G_t$. Let h=e(g,g).
  - Alice picks a random key a. Bob picks b, Carol picks c.
  - Alice broadcasts $g^a$, Bob broadcasts $g^b$, Carol broadcasts $g^c$.
  - Alice computes $(e(g^b,g^c)^a = h^{abc}$. Bob and Carol compute $h^{abc}$ similarly.

# Security

- The bilinear mapping lets Alice computes $h^{bc}$ from $g^b$ and $g^c$, and then raises it to the power of a.
- An external adversary cannot compute $h^{abc}$ from $g^a, g^b, g^c$.
  - Cannot compute $e(g^a, e(g^b, g^c))$, since $e(g^b, g^c)$ is in $G_t$ and not in G.
  - This is the Bilinear Diffie-Hellman assumption. (We need the Decision Bilinear Diffie-Hellman assumption which states that it is impossible to distinguish $h^{abc}$ from random.)

# The setting

- Key generation center (KGC)
  - Holds the master private key
  - Generates public system parameters
- Key derivation: The KGC can provide each user with the private key corresponding to his/her name.
  - The private key is a function of the name (or an arbitrary string) and the master private key
- Encryption: everyone can encrypt messages to Alice. The ciphertext is a function of the plaintext, Alice's name, and the public parameters.
- Decryption: Alice uses her private key and the system parameters to decrypt messages sent to her

# Boneh and Franklin's IBE scheme

- Let G be a group of order q in which DH is hard, and g a generator of G. $e: G \times G \rightarrow G_t$.
  - Let $h=e(g,g)$.
  - Let $H_1: \{0,1\}^* \rightarrow G$, and $H_2: G_t \rightarrow \{0,1\}^*$ be two hash functions.
- Setup:
  - KGC picks a random $s \in [1,q]$. $g^s$ is the public key.
- Private Key:
  - The KGC gives Bob the private key $H_1(Bob)^s$.

# Boneh and Franklin's IBE scheme

- Encryption:
  - To send m to Bob, pick $r \in [1,q]$.
  - Ciphertext = $(g^r, m \oplus H_2( e( H_1(Bob),g^s )^r ))$
    $= (g^r, m \oplus H_2( e(H_1(Bob),g)^{rs} ))$

- Decryption:
  - Bob has an encrypted message (u,v) and a private key $w=H_1(Bob)^s$.
  - He computes $v \oplus H_2(e(w,u)) = m \oplus H_2( e(H_1(Bob),g)^{rs} ) \oplus H_2(e(H_1(Bob)^s, g^r)) = m$.

# Boneh and Franklin's IBE scheme

- Intuition:
  - The message is encrypted with $H_2( e(H_1(Bob),g)^{rs} )$
  - Similar to 3-party DH where
    - The sender has public key $g^r$, private key r.
    - The KGC has public key $g^s$, private key s.
    - The recipient has public key $H_1(Bob)$, no private key.
    - The session key is $H_1(Bob)^{rs} = h^{rs \log(H(Bob)}$.
  - But the KGC gives $H_1(Bob)^s$ to the recipient, so he can use it to find the session key.
  - The security proof assumes that $H_1,H_2$ are random oracles

# BLS signature scheme

- Boneh, Lynn and Shacham gave a simple, deterministic signature scheme based on pairings.
  - The signatures are very short.
  - Security is proven under the random-oracle model.
- Keys:
  - Private key: x. Public key: $g^x$. Hash function $H() \rightarrow G$.
- Signature:
  - $Sign(m) = \sigma = (H(m))^x$ (in G).
- Verification:
  - Check if $\langle g, g^x, H(m), \sigma \rangle$ is a DDH tuple. Namely, check if $e(g,\sigma)=e(g^x,H(m))$.

# BLS signature scheme

- Security:
  - Unexistentially forgeable
  - under adaptive chosen message attack
  - in the random oracle model
  - assuming that the CDH is hard on certain elliptic curves over a finite field of characteristic
- Efficiency:
  - signing is fast, one hashing operation and one exponentiation.
  - Verification requires two pairing computations,
  - The signature is just an element in G, which is 154 bits long if we use an elliptic curve on $F_{3^{97}}$
  - half the size of DSA (El Gamal variant) signature in DSA (320 bits) with comparable security.

# Multisignature

- Several signers need to sign the same message m.
- Each signer $P_i$ has secret key is Xi and public key Yi = $g^{Xi}$.
- Signature: the signature on m is $\sigma = \Pi_{i=1,\ldots,n}\sigma_i$, where $\sigma_i$ is the BLS signature. Namely, each signer computes $\sigma_i=(H(m))^{Xi}$ and then they multiply their signatures.
- Verification:
  - As in BLS, accept if $e(g,\sigma) = e(\Pi_{i=1,\ldots,n} Yi, H(m))$

# Aggregate signatures

- Several signers want to sign different message $m_1,\ldots,m_n$. (e.g., certificates.)
- Each signer $P_i$ has secret key is Xi and public key Yi = $g^{Xi}$.
- Siganture:
  - Frst, each signer computes its signature $\sigma_i=(H(m_i))^{Xi}$
  - The signers then multiply their signatures, $\sigma= \Pi_{i=1,\ldots,n}\sigma_i$.
- Verification:
  - Accept if $e(g,\sigma) = \Pi_{i=1,\ldots,n}\ e(Yi, H(m_i))$
- This scheme is secure against existential forgery with chosen message attacks if the computational Co-DH problem is hard: given g, $g^a$ (in G), and h (in $G_t$), it is hard to compute $h^a$ in $G_t$.